# ByteCTF Crypto

原创

PUTAOAO 于 2021-10-18 21:57:51 发布 57 收藏

分类专栏： 刷题记录 文章标签： Crypto



本文链接：https://blog.csdn.net/PUTAOAO/article/details/120835426





## easyxor

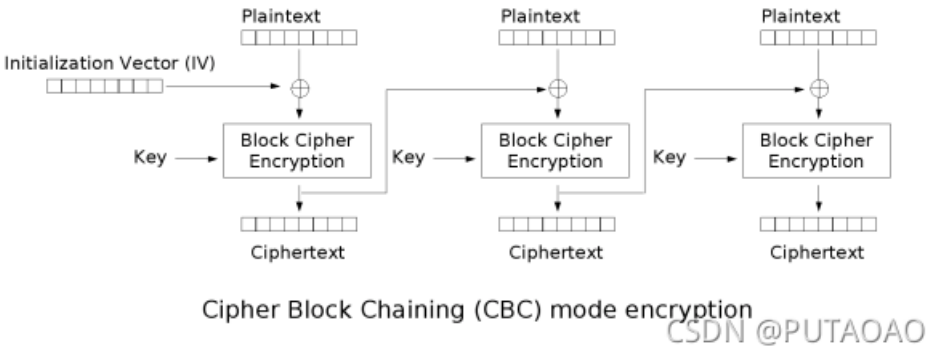
AES的CBC模式和OFB模式

之前赵师傅讲过 听得有点云里雾里的

现在跟着这题的wp顺一遍

首先iv与key是不知道的，只知道前半段密文是OFB模式，后半段是CBC模式

贴两张图



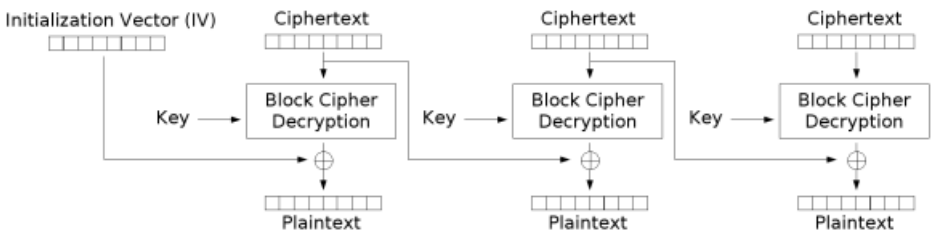

Cipher Block Chaining (CBC) mode encryption



这个是CBC加密，我们可以知道最后一个密文的iv就是前一个密文

通过这个可以爆破出key 因为我们知道flag最后是以}$$$$结尾的

**这个解密图里的iv就是加密的iv 密文顺序也是一样的而给的wp是逆序解的**自己改成正序了




Cipher Block Chaining (CBC) mode decryption



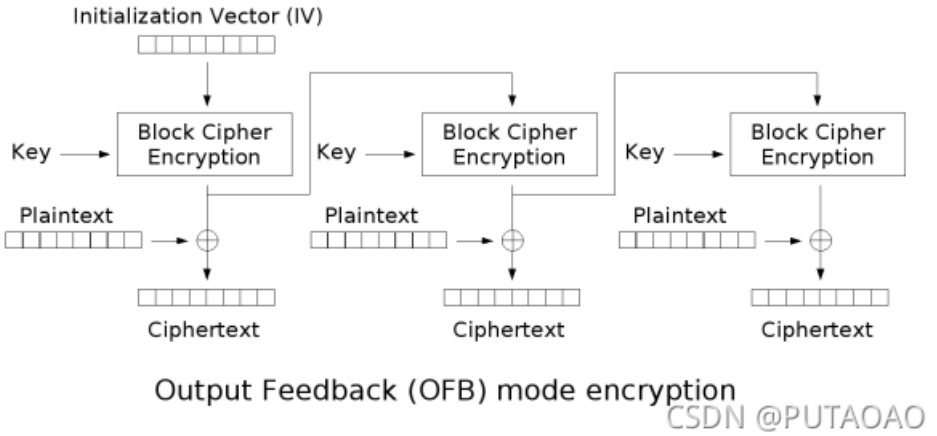
```
from Crypto.Util.number import *
# right shift inverse
def inverse_right(res, shift, bits=64):
    tmp = res
    for i in range(bits // shift):
```

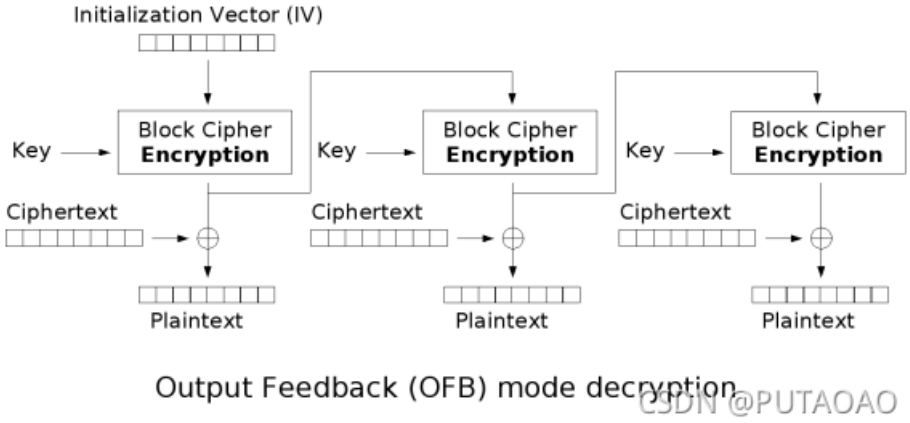
```
        tmp = res ^ tmp >> shift
    return tmp


# right shift with mask inverse
def inverse_right_mask(res, shift, mask, bits=64):
    tmp = res
    for i in range(bits // shift):
        tmp = res ^ tmp >> shift & mask
    return tmp

# left shift inverse
def inverse_left(res, shift, bits=64):
    tmp = res
    for i in range(bits // shift):
        tmp = res ^ tmp << shift
    return tmp


# left shift with mask inverse
def inverse_left_mask(res, shift, mask, bits=64):
    tmp = res
    for i in range(bits // shift):
        tmp = res ^ tmp << shift & mask
    return tmp

def unshift(m, k, c):
    if k < 0:
        return inverse_right_mask(m,-k,c)
    return inverse_left_mask(m,k,c)
def unconvert(m, key):
    c_list = [0x37386180af9ae39e, 0xaf754e29895ee11a, 0x85e1a429a2b7030c, 0x964c5a89f6d3ae8c]
    for t in range(3,-1,-1):
        m = unshift(m, key[t], c_list[t])
    return m
def convert(m, key):
    c_list = [0x37386180af9ae39e, 0xaf754e29895ee11a, 0x85e1a429a2b7030c, 0x964c5a89f6d3ae8c]
    for t in range(4):
        m = shift(m, key[t], c_list[t])
    return m
def shift(m, k, c):
    if k < 0:
        return m ^ m >> (-k) & c
    return m ^ m << k & c
def decrypt(c, k, iv, mode='CBC'):
    cipher=[]
    for i in range(0,len(c),16):
        cipher.append(int(c[i:i+16],16))
    groups=[]
    if mode == 'CBC':
        i=0
        for eve in ([iv]+cipher[:-1]):
            cur_c = cipher[i]
            cur = unconvert(cur_c, k)
            groups.append(cur ^ eve)
            i+=1
    elif mode == 'OFB':
        last = iv
        for eve in cipher:
            cur_c = convert(last, k)
```

```
            cur_c = convert(last, k)
            groups.append(cur_c ^ eve)
            last = cur_c
    else:
        print('Not supported now!')
    m = b''
    for i in groups:
        m += long_to_bytes(i)
    return m
def get_key():
    for a in range(-32,33):
        for b in range(-32,33):
            for c in range(-32,33):
                for d in range(-32,33):
                    try:
                        plain = decrypt(c2[-16:],[a,b,c,d],c_list[-2])
                        if(plain.endswith(b'$$$') and plain.strip(b'$').endswith(b'}')):
                            print(a,b,c,d,plain)
                            return [a,b,c,d]
                    except:
                        continue
cipher1='89b8aca257ee2748f030e7f6599cbe0cbb5db25db6d3990d3b752eda9689e30fa2b03ee748e0da3c989da2bba657b912'
flag=b'ByteCTF{'
m=bytes_to_long(flag)
c1,c2=cipher1[:len(cipher1)//2],cipher1[len(cipher1)//2:]
c_list = []
for i in range(0,len(c2),16):
    c_list.append(int(c2[i:i+16],16))
get_key()
```

然后再通过OFB模式得到iv

下面是OFB加密和解密（好图）




我们可以知道flag形式为ByteCTF{开头
刚好8位
这个就是plaintext而ciphertext是知道的
两个异或一下就是key和iv的加密
逆一下就能得到iv

```
c_list = []
for i in range(0,len(c1),16):
    c_list.append(int(c1[i:i+16],16))
E = (m ^ c_list[0])
iv = unconvert(E, key)
#print(iv)
```

最后解密

```
c_list = []
for i in range(0,len(c1),16):
    c_list.append(int(c1[i:i+16],16))
E = (m ^ c_list[0])
iv = unconvert(E, key)
#print(iv)
m1=decrypt(c1,key,iv,mode='OFB')
m2=decrypt(c2,key,iv)
print(m1+m2)
```

总的代码

```
from Crypto.Util.number import *
```

```python
# right shift inverse
def inverse_right(res, shift, bits=64):
    tmp = res
    for i in range(bits // shift):
        tmp = res ^ tmp >> shift
    return tmp


# right shift with mask inverse
def inverse_right_mask(res, shift, mask, bits=64):
    tmp = res
    for i in range(bits // shift):
        tmp = res ^ tmp >> shift & mask
    return tmp

# left shift inverse
def inverse_left(res, shift, bits=64):
    tmp = res
    for i in range(bits // shift):
        tmp = res ^ tmp << shift
    return tmp


# left shift with mask inverse
def inverse_left_mask(res, shift, mask, bits=64):
    tmp = res
    for i in range(bits // shift):
        tmp = res ^ tmp << shift & mask
    return tmp

def unshift(m, k, c):
    if k < 0:
        return inverse_right_mask(m,-k,c)
    return inverse_left_mask(m,k,c)
def unconvert(m, key):
    c_list = [0x37386180af9ae39e, 0xaf754e29895ee11a, 0x85e1a429a2b7030c, 0x964c5a89f6d3ae8c]
    for t in range(3,-1,-1):
        m = unshift(m, key[t], c_list[t])
    return m
def convert(m, key):
    c_list = [0x37386180af9ae39e, 0xaf754e29895ee11a, 0x85e1a429a2b7030c, 0x964c5a89f6d3ae8c]
    for t in range(4):
        m = shift(m, key[t], c_list[t])
    return m
def shift(m, k, c):
    if k < 0:
        return m ^ m >> (-k) & c
    return m ^ m << k & c
def decrypt(c, k, iv, mode='CBC'):
    cipher=[]
    for i in range(0,len(c),16):
        cipher.append(int(c[i:i+16],16))
    groups=[]
    if mode == 'CBC':
        i=0
        for eve in ([iv]+cipher[:-1]):
            cur_c = cipher[i]
            cur = unconvert(cur_c, k)
            groups.append(cur ^ eve)
            i+=1
```

```
    elif mode == 'OFB':
        last = iv
        for eve in cipher:
            cur_c = convert(last, k)
            groups.append(cur_c ^ eve)
            last = cur_c
    else:
        print('Not supported now!')
    m = b''
    for i in groups:
        m += long_to_bytes(i)
    return m
def get_key():
    for a in range(-32,33):
        for b in range(-32,33):
            for c in range(-32,33):
                for d in range(-32,33):
                    try:
                        plain = decrypt(c2[-16:],[a,b,c,d],c_list[-2])
                        if(plain.endswith(b'$$$') and plain.strip(b'$').endswith(b'}')):
                            print(a,b,c,d,plain)
                            return [a,b,c,d]
                    except:
                        continue
cipher1='89b8aca257ee2748f030e7f6599cbe0cbb5db25db6d3990d3b752eda9689e30fa2b03ee748e0da3c989da2bba657b912'
flag=b'ByteCTF{'
m=bytes_to_long(flag)
c1,c2=cipher1[:len(cipher1)//2],cipher1[len(cipher1)//2:]
c_list = []
for i in range(0,len(c2),16):
    c_list.append(int(c2[i:i+16],16))
key=[-12,26,-3,-31]
c_list = []
for i in range(0,len(c1),16):
    c_list.append(int(c1[i:i+16],16))
E = (m ^ c_list[0])
iv = unconvert(E, key)
#print(iv)
m1=decrypt(c1,key,iv,mode='OFB')
m2=decrypt(c2,key,iv)
print(m1+m2)
```

~~好像还有点小问题，之前想通了，现在又想不通了，明天再改~~